Keywords

- and - Boolean operator, only returns True when both conditions are ‘True’
- del - deletes objects (items in a list)
- from - Opening part of an import statement
- not - Boolean operator, reverses ‘True’ or ‘False’ outcomes
- while - Continuous loop - runs until statement returns False
- as - Part of an import statement - imports a module under a different name
- elif - “Else if” Allows additional if gates to be added to ‘if’ statements
- global - Allows variables within functions to amend values of that variable outside the function
- or - Boolean operator, returns True if either condition is True
- with - works with file streams, automatically closes the file stream and ‘cleans’ the code when the code ‘nested’ under it finishes
- assert - Used to test conditions in a program that should never happen
- else - Closes an ‘if’ statement, if none of the ‘if’ statement conditions are met runs the code nested underneath
- if - starts an ‘if’ statement. If the condition it specifies is true, runs the code nested underneath
- pass - does nothing, used to skip functions that haven’t been written yet
- yield - used with generators, exits the generator and returns a value
- break - used to interrupt a cycle.
- except - part of a ‘try’ statement. If an exception is raised it executes code nested underneath
- import - part of an import statement imports a specific function
- print - displays on screen
- class - creates new user defined objects (important!)
- exec - executes python code dynamically
- in - part of a for statement, tests to see if a value exists in a list
- raise - creates a user defined exception
- continue - interrupts the current cycle without jumping out of the whole cycle a new cycle will begin
- finally - cleans up resources at the end of a try statement
- is - boolean operator, tests for object identity
- return - returns a value after a function
- def - start of a function statement
- for - starts a for loop
- lambda - creates an anonymous function - a function not bound to a specific name
- try - used to handle exceptions, if no exception is raised it ends. If one is then it runs the except code underneath

Data Types

For data types, write out what makes up each one. For example, with strings write out how you create a string. For numbers write out a few numbers.

- True - Boolean data type, often used to check a condition
- False - “”
- None - Created by a pass keyword in a function. Stands for non-existant data
- strings - Data type representing textual data, created by enclosing text in “”, ‘’ or “”” “””
  quotes
- numbers - Can be integers, floating point values or complex numbers. Integers are numbers with no decimals. E.g. 5
- floats - represent ‘real’ numbers in computing e.g. with a decimal (5.5)
- lists - created by enclosing data seperated by commas in square brackets [] - e.g. [1, 2, 3, 4, 5] can contain numeric or textual data.

String Escape Sequences

For string escape sequences, use them in strings to make sure they do what you think they do.

- \\ - allows you to include a backslash
- \' - single quote
- \" - double quote
- \a - ASCII Bell (originally used to send a ring on a ticker tape)
- \b - backspace
- \f - ASCII formfeed may be present at the start of a line and will be ifgnored for nesting purposes
- \n - new line
- \r - carriage return
- \t - indents text (tab)
- \v - vertical tab

String Formats

Same thing for string formats: use them in some strings to know what they do.

- %d - integer
- %i - integer
- %o - octal value
- %u - identical to d (integer)
- %x - hexadecimal value (lowercase)
- %X - hexadecimal value (uppercase)
- %e - floating point exponential format (lowercase)
- %E - floating point exponential format (uppercase)
- %f - floating point decimal format (lowercase)
- %F - floating point decimal format (uppercase)
- %g - Floating point format. Uses lowercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.
- %G - Floating point format. Uses uppercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.
- %c - Single character (accepts integer or single character string).
- %r - String (converts any Python object using *repr()*).
- %s - String (converts any Python object using str()).
- %% - No argument is converted, results in a '%' character in the result.

Operators

Some of these may be unfamiliar to you, but look them up anyway. Find out what they do, and if you still can't figure it out, save it for later.

- + - adds two values
- -   - subtracts values
- * - multiplies values
- ** - to the power of
- / - division
- // - integer division
- % - modulo
- < - less than
- > - more than
- <= - less than or equal to
- >= - more than or equal to
- == - equal
- != - not equal
- <> - not equal
- ( ) - holds arguments
- [ ] - holds a list
- { } - holds a range
- @ - class function and object decorators
- , - concatenates a string, separates a list
- : - symbolises the end
- . - runs a function
- = - sets a variable
- ; - sets a new line in python, you can run multiple lines of code in one line
- += - add an assignment operator (sets a variable and adds)
- -= - subtract and assignment
- *= - multiply and assignment
- /= - divide and assignment
- //= - integer divide and assignment
- %= - modulo and assignment
- **= - power of and assignment